International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:2, 2010

Completion Latin Square for Wavelength Routing

Ali Habiboghli, Rouhollah Mostafaei, and Vasif Nabiyev

***Abstract*—**Optical network uses a tool for routing called Latin router. These routers use particular algorithms for routing. For example, we can refer to LDF algorithm that uses backtracking (one of CSP methods) for problem solving. In this paper, we proposed new approached for completion routing table (DRA&CRA algorithm) and compare with pervious proposed ways and showed numbers of backtracking, blocking and run time for DRA algorithm less than LDF and CRA algorithm.

***Keywords*—**Latin Router, Constraint Satisfaction Problem, Wavelength Routing.

I. Introduction

THE constraint paradigm is a useful and well-studied framework expressing many problems of interest in Artificial Intelligence and other areas of Computer Science. A significant progress has been made in the last decade. Many real-life problems can be expressed as a special case of the constraint satisfaction problem. Some examples are scheduling, configuration, hardware verification, graph problems, molecular biology, Optical network routing [8] etc. The search space is often exponential because the problem is NP-complete. Therefore a number of different approaches to the problem have been proposed to reduce the search space and find a feasible solution in a reasonable time.

A CSP is defined by a finite set of problem variables along with their associated finite domains of possible values and a set of constraints on acceptable combinations of the values. A constraint can be given either explicitly, by enumerating the allowed combinations, or implicitly, e.g., by an algebraic expression.[6]

A constraint satisfaction problem (CSP) is a triple $P=(X,D,C)$, where

- $X=\{x_1,\ldots,x_n\}$ is the set of variables called variables;
- $D=\{D_1,\ldots,D_n\}$ is the set of domains. Each domain is a finite set containing the possible values for the corresponding variable;
- $C=\{C_1,\ldots,C_c\}$ is the set of constraints. A constraint C_i is a relation defined on a subset $\{x_{i1},\ldots,x_{ik}\}$ of all variables, i.e., $C_i \subseteq \{D_{i1}\times\ldots\times D_{ik}\}$. The set of variables $\{x_{i1},\ldots,x_{ik}\}$ is referred to as a scope of the constraint C_i.

Finally, constraint satisfaction problem (CSP) distributes the problem and divides it in to three topples that defined as $P=(X,D,C)$ and solves the problem. In optical network, Latin routers have been used for wavelength routing that these routers can be solved by CSP. An example of these algorithms is LDF [1], in which by growing the degree of density, blocking probability increases too. In this paper, for solving this problem, we propose new algorithms and compare it with pervious algorithm.

The framework of this paper is as follow: in section II we introduce a general description of the problem. In section III our pervious proposed algorithm has been discussed. In section IV we introduce our new proposed algorithm and finally we show our experimental results about this problem.

II. Problem Description

Consider a wavelength routing device with N input ports and N output ports. A Latin Square, L, is a routing table in the form of an $N\times N$ matrix that specifies the wavelength connections from the N input ports to the N output ports such that there is no wavelength conflict in any of the N output ports (it is assumed that the wavelengths arriving at any input port are all distinct).[2][7]

The matrix contains elements from a finite set $S=\{1,2,3...,n\}$ (representing the set of available wavelengths) such that each row and each column of the matrix contains all the elements of S, and, none of the rows and columns contains the same element more than once (see Fig. 1(a) for an example). An $s_{ij}\subset S$ in the (i, j) entry, l_{ij} of the matrix L implies that the wavelength s_{ij} from input port I will be routed to output port j.[3][4] Formally, a $N\times N$ Latin Square, L, is defined as:

Ali Habiboghli is with the Islamic Azad University of KHOY Branch, Iran. He is now with the department of computer engineering (e-mail: habiboghli@iaukhoy.ac.ir, ali.habibogli@gmail.com).

Rouhollah Mostafaei is with the Islamic Azad University of KHOY Branch, Iran. He is now with the department of computer engineering (e-mail: mostafaevn@iaukhoy.ac.ir, rk_mostafa@yahoo.com).

Vasif Nabiyev is with the Karadeniz Technical University Trabzon, Turkey. He is now with the Department of Computer Engineering (e-mail: vasif@ktu.edu.tr).

$$L=[L_{ij}] \quad , L_{ij} \in S=[1,2,3,.....,N] \quad ,i,j=\{1,2,3,.....,N\}$$
$$L_{ij} \neq L_{ik} \quad \forall i,j,k \quad j \neq k$$
$$L_{ji} \neq L_{ki} \quad \forall i,j,k \quad j \neq k \qquad (I)$$

A $N \times N$ Partial Latin Square (PLS) is an $N \times N$ matrix in which some of the entries can be empty (φ) or unassigned. However, each of the assigned entries must contain an element of S satisfying (I). Density of a PLS is defined as the ratio of the number of assigned entries to the total number of entries.

Formally, an N×N Partial Latin Square, LP, is defined as:

$$L_p=[L_{ij}] \quad , L_{ij} \in S=[\phi,1,2,3,.....,N] \quad ,i,j=\{1,2,3,.....,N\}$$
$$\forall L_{ij} \neq \phi: \quad L_{ij} \neq L_{ik} \quad \forall i,j,k \quad j \neq k$$
$$\forall L_{ij} \neq \phi: \quad L_{ji} \neq L_{ki} \quad \forall i,j,k \quad j \neq k \qquad (II)$$

If each of the empty entries of a PLS can be assigned an element from S without violating (1), then we say that there exists a completion of the PLS. However, not every PLS can be completed (e.g., PLSs in Fig. 1(b)). In Fig. 1(b), there is no way to assign the Wavelength 1 into an empty entry without violating (II) (note that, each wavelength should occur N=4 times in the completed matrix) [1].

III. Description of Previous Proposed Algorithm

Two algorithms have been developed to complete a given PLSs. Algorithm 1 is based on the *backtracking* method with heuristics to minimize backtracking. Algorithm 2 guarantees a completion of the PLS by employing a *maximum weighted matching* algorithm for bipartite graph, however, some of the reassigned entries in the PLS may be reassigned (equivalent to wavelength conversion). Here, we describe first algorithm.

A general way to complete a PLS is backtracking method, one of the most commonly used techniques to systematically exhaustive search for a set of solutions satisfying certain constraint. However, the number of alternative large for sparse PLSs, and the hence complexity of backtracking can grow exponentially to reduce the number backtracking steps a heuristic used which is based on the concept of degree of freedom[20]. Degree of freedom (DOF) for each empty entry in a PLS is defined as the number of elements that can be assigned in to that entry without violating property (2). Degree of freedom for each assigned entry is defined to be zero. For example, degree of freedom for each of the elements of the PLS in Fig.2 (a) is shown in Fig.2 (b).

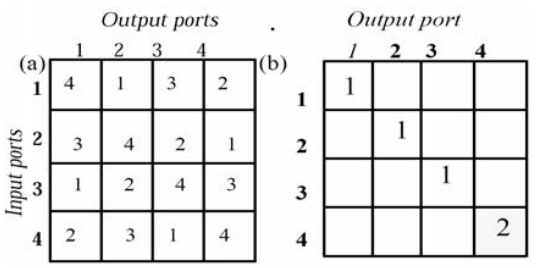

Fig. 1(a) Examples of a 4x4 Latin Square
(b) Partial Latin Squares w/o any completion

Each assignment of an empty entry may reduce the degrees of freedom of some of the entries sharing the same row or the same column and can even *block* those entries. *E.g.*, if Wavelength 3 is assigned to (2, 3), *DoF* of entries (2, 4) and (3,3) will be reduced by 1. It's thus intuitive that the empty entries should be assigned in the increasing order of their degrees of freedom. If the DoF of the entry selected to be assigned next is one then we have no choice but to assign a predetermined wavelength into that entry.

Otherwise *(i.e.*, if DoF > 1), a valid wavelength is assigned that results in the minimum reduction in the total degrees of freedom of the entries in the same row and in the same column. *E.g.*, if 3 is assigned to (2, 3) then the total reduction in DoF of the entries in the Row 2 and Column 3 (excluding entry (2,3)) is 2. However, if 1 is assigned to (2, 3) then the total reduction in DoF is 3. Hence, Wavelength 3 will be assigned to (2, 3) if it is chosen as the next entry to be assigned. Then, the DoF of all the affected entries are updated. Now the algorithm checks if any entry has zero DoF (*i.e., blocked*). If there is a blocked entry then the algorithm will backtrack to the previously assigned entry, cancel that assignment and will reassign that entry with a wavelength that has not been considered before and which results in the minimum reduction in the total DoF. Otherwise, if none of the entries end up with zero DoF, the algorithm selects the next entry with minimum DoF and repeats the above procedure. The algorithm stops when all the *N*2 entries are assigned or when it identifies that the PLS cannot be completed.

IV. Proposed Algorithm

A. Description of CRA Algorithm

LDF algorithm and its operation have been described above. Because of increasing density in this algorithm, the number of steps of backtracking increases too. (It means that with increasing the ratio of the number of predefined entities to N^2, average number of backtracking increase too). For this reason we introduce new algorithm that improves pervious LDF algorithm. For more clarity we will explain this algorithm later. We call it, 'Constraint Reduction Algorithm'.
to understand completing Latin Square with constraint reduction algorithm see Fig. 2. Every row in Latin Square has separate domain of values, which should be assigned for entries. For a Latin Square with size of $N \times N$, the number of elements domain is N that this set of domain equals to: $\{1,2,3\ldots,n\}$.

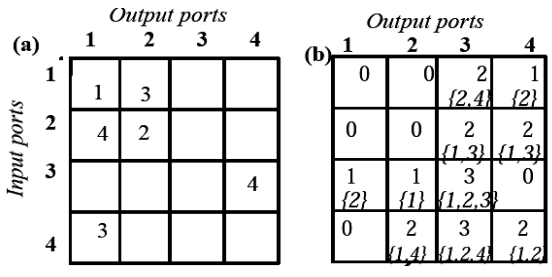

Fig. 2 (a) A Partial Latin Square
(b)Corresponding degrees of freedom

A number is selected from value domain and assigned to an entity; it should be viewed as a constraint in the value domain of next rows in this column. It means that we can't assign this number for that entity. For example if we assign value1 to entity (1,1), then we can't assign it to entity (2,1).

Each value that is assigned from domain of one row to entities of that row should be deleted from domain. For completing the next row, each candidate value is compared with invalid values, when they are not equally assigned for entities. For example, if value 1 is selected from domain of second row, we can not assign it to entity (2, 1) because 1=1 and so next value should be tried. The way of completing entities of other rows is similar.

Now the algorithm there for proposed algorithm use constraint reduction for solving LS and perform as well as explained in Fig. 3. We continue by implementing LDF and proposed algorithm and compare both of them. Our comparisons include execution cost and number of backtracks. We will show our results in the charts.

```
procedure CRA
Begin
repeat for I = 1 to number of rows
        repeat for each row until ele element of that row satisfied
                – set domain of each cell
                – select in domain of row and pick to cell
                  where condition is satisfied
                – each number that pick in cells
                  delete in same collomn for next rows
        Endfor
 Endfor
End
```

Fig. 3 CRA algorithm for solving Latin Square problem

```
procedure  DRA
repeat  for number  of  rows
    repeat  for each rows untile  S_i is not null
        select  Random  in  S_i
        if  item  exist  in domain  S_j
        where  coresspond  ing ofassignme  nt of  cell then
        Begin
            – assignment  the  item in cell  of  latin  square
            – the item assignment  geting  in stack
            – delete  item in  S_i and  S_j coresspond  ing cell
        End
    else
        Begin
            – pick  the  frist  item in stack
            – frist  item  add  in S_i and  S_j previous  cell .
        End
    Endfor
Endfor
```

Fig. 4 DRA algorithm for solving Latin Square problem

B. Description of DRA Algorithm

Constraint satisfaction problem is based on test and generate. Furthermore CSP for solving problem has more search space. Therefore, for problem that search space is more, we can reduce the extra state. In DRA algorithm, we use it. For solving problem, suppose relation I. entity of LS should be complete where rows and columns are not same element. Therefore, the number of domains for assign is equal 4(i.e. domain is equal {1, 2, 3, and 4}).

For complete Latin square, we complete first row and then next rows respectively. Therefore, we select a value from domain of first row and assign in cell (1, 1) where a selected value exists in domains to column1 i.e. Selected value from row domain should exist in column then value is assigned in cell otherwise algorithm should be backtracked and select the other value and try. For example, if we assign 3 in cell (2, 3) then after assigning 3 is deleted from domain of row 2 and also 3 are deleted from domain of column 3. When the algorithm backtracks, domain should be updated. For other rows, this work tries. DRA algorithm has been shown in Fig. 4.

V. Experimental Result

We implement proposed algorithm and both previous ways. We test our algorithm by 100 times of each LS with size variation from 1 to 10 and evaluate two algorithm. In different executions of algorithm, we obtain the number of backtracks in each state and average number of backtrack in each algorithm. As shown in Fig. 5, for increasing the size of LS, the number of backtracks increase too.

Test of different executions show that LDF and CRA algorithm has more backtracks than our proposed algorithm. In these experiments we obtained the ratio of execution time and different size of LS for CRA, LDF and DRA algorithm. With attempting Fig. 6, we can see that our proposed algorithm from the execution time perspective is better than CRA and LDF algorithm.

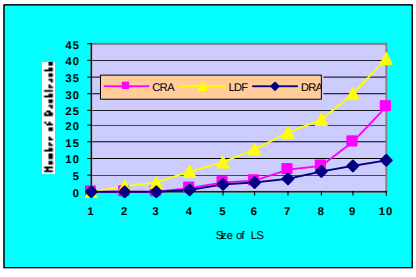

Fig. 5 Average number of backtracking for ascending size of LS

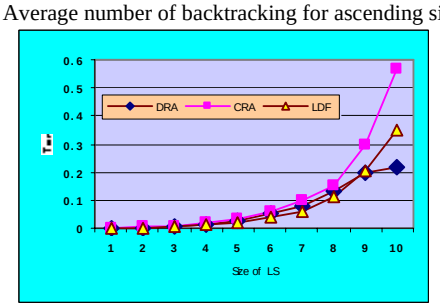

Fig. 6 Average runtime for ascending size of LS

The advantage of both algorithms is that in any way they can fined the solution (solve the problem). It means that after executing, both algorithms we complete LS. As we can see in Fig. 5, the average number of backtrack in CRA algorithm is equal to 30, however this value for LDF is 40 and for DRA equal 10. Furthermore, LDF algorithm has blocking probability; on the other hand, deadlock may occur in LDF. Table I shows the number of executions and blocking percent for both algorithms.

TABLE I
PERCENT OF BLOCKING

Operation	CRA	LDF	DRA
Number of block	0	637	0
Total execution	300	2490	300
Percent of blocking	%0	%25.58	%0

VI. CONCLUSION

In this paper, we introduce new approaches for completion Latin Square. This DRA method has more performance than LDF and CRA algorithm. CRA and DRA algorithm are never blocked, but the LDF algorithm blocked with increasing degree of density and number of backtracking in our proposed algorithms is less than LDF algorithm. In the next work, we will use these algorithms in Latin Router for routing wavelength.

ACKNOWLEDGMENT

I would like to thank my advisor, Arthur Vasif Nabiyev for all his help and Hale Mojarrabi for editing this work.

REFERENCES

[1] C. Chien and S. Banerjee. “Optical switch configuration and lightpath assignment in wavelength routing multihop lightwave networks.” Proceedings of the 14th IEEE Info COM, 1995
[2] D.Banerjee and J.Frank, “Constraint Satisfaction in Optical Routing for Passive Wavelength-Routed Networks”, Department of Computer Science University of California, 1998.
[3] D.Banerjee, J.Frank, and B.Mukherjee, “Passive Optical Network Architecture Based on Waveguide Grating Routers”, Department of Computer Science University of California, March 1, 1996.
[4] D. Banerjee and J. Frank, and B. Mukherjee, “Passive Optical Network Architecture Based on Waveguide Grating Routers” IEEE Journal on selectet areas in communications, september 1998.
[5] M. H. Saqalli, L.Purvis and E.C.Freuder “Survey of Application Integrating Constraint Satisfaction and Case-Based Reasoning” university of New Hampshire, pages3-4, 2000.
[6] K.Vermirovsky, "Algorithms for Constraint Satisfaction Problems", Master Thesis of Purdue University, 2003.
[7] Carla P. Gomes and D.hmoys. “Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem.” In Proc. Computational Symposium on Graph Coloring and Generalizations, Cornell University, Dept. of Comp. Science, 2002.
[8] Habiboghli, A., Mostafaei, R., and Meybodi, M. R., "Using Latin Router for Routing Wavelength with Configuration Algorithm", Proceedings of the WCSET 2009: World Congress of Science, Engineering and Technology, Venice, Italy, pp. 61-64 July 29-31, 2009.

Ali Habiboghli received the B.S. degree from department of engineering of Islamic Azad University, KHOY Branch in 2004. He received the M.S degrees from electronic, computer engineering and information technology from Islamic Azad University of Qazvin Branch, Iran in 2007. From 2007 to already is with the Islamic Azad University of KHOY Branch. His research focuses on artificial intelligence, algorithms, distributed systems.

Rouhollah Mostafaei received the B.S. degree from department of engineering of Islamic Azad University, KHOY Branch in 2004. He received the M.S degrees from electronic, computer engineering and information technology of Islamic Azad University, Qazvin Branch, Iran in 2007. From 2007 to already is with the Islamic Azad University of KHOY Branch. His research focuses on artificial intelligence, computer networks, distributed systems.